Четвертое практическое домашнее задание.
Дедлайн: 16.04.2023 23:59:59

Давайте подумаем, что такое сеть? Очевидно, сеть это надстройка над графом, которая хранит какие-то свойства для ребер, например, поток через них и их емкость. Значит ли это, что под сеть надо менять граф? Очевидно нет! Таким образом, требования к графу остаются теми же. Но как же хранить эти величины, не меняя ребер? Давайте в сети хранить помимо графа отображение из ребра в структуру, хранящую свойства.

С потоками разобрались, а что с остаточной сетью? Предлагается создать класс `FilteredGraph`, в котором сохраняются только ребра, удовлетворяющие некоторому предикату, который можно передать в конструктор. Например, для остаточной сети нужно будет поддержать ребра с ненулевой остаточной емкостью. Как это реализовать? Внутри `FilteredGraph` достаточно хранить указатель на граф, а при получении списка соседей вернуть вектор из тех, ребра до которых удовлетворяют предикату (в качестве предиката удобно использовать механизм лямбда-функций. Аналогично решается проблема с слоистой сетью.

В алгоритме Форда-Фалкерсона вы используете DFS, казалось бы, у нас есть DFS с первого контеста, так что на сцену выходит ваш любимый визитор! Он должен уметь сохранить путь, вдоль которого вы пропустите поток. Тогда алгоритм Форда-Фалкерсона это по сути запуск DFS, извлечение пути из визитора, вычисление дополнительного потока вдоль него и пропускание его по сети.

Казалось бы, мы решили все проблемы, но нет. В общем-то граф имеет методы добавь вершину, добавь ребро, и так далее. Однако в сети гораздо больше параметров, надо присваить стоки и истоки, к каждому ребру сразу добавлять обратное и около того. Тем более не хочется менять класс сети, чтобы строить новую сеть. Для такого случая применяют паттерн `builder`, который заключается в том, что есть какой-то класс-строитель, который строит сеть за него путем добавления ребер, присваивания стока, истока и так далее. Однако в общем случае это необязательное требование, если вам удастся грамотно построить логику построения сети.

Таким образом, получаем список требований к сети.

1. Требования к графу остаются неизменными, для использования его в сети заведите отображение из ребра в структуру с его потоковыми свойствами.
2. Для получения остаточной сети заведите обертку над графом, чьи ребра удовлетворяют некоторому предикату.
3. Код DFS/BFS не должен меняться. Используйте визиторы.
4. (Опционально.) Можно реализовать `NetworkBuilder`, чтобы вынести сложную логику из конструктора.
5. У вас должен быть способ получить минимальный разрез с минимальным изменением кода.